PAGE FLIPPING

Page Flipping is a set of simple programs designed to teach those new to the ATARI how to store information in memory and then bring it back on the screen instantly! With the two methods taught here, even a new programmer can learn to do simple animation, or present nice slide-like displays. The simplest of these examples has only 12 short lines of basic code!

The person using this lesson should be familiar with BASIC programming so that he or she can read the code that is included. Since the program is not protected, the user is encouraged to try their own modifications inorder to gain a greater perspective of the art of flipping pages of memory.

The program is split up into smaller pieces that will load and run on ATARI(tm) 400/800 computers with 16k RAM for cassette users and 24k RAM for those using disk.

Educational Software inc.
4565 Cherryvale
Soquel, CA 95073
(408) 476-4901

Educational Software

presents

TRICKY TUTORIAL #3

PAGE FLIPPING

TRICKY TUTORIAL #3

PAGE FLIPPING

by
Robin Sherer

No doubt some of you hesitated before ordering a program to "flip pages". For this reason I want to start by assuring you that once you learn either of the two simple methods taught here, your programs will look and run much nicer.

Not very long ago, I had to learn the method of flipping screens. Since I still remember how confusing it was at first, only the material you need is presented here. A few details have been left out because only those trying to become experts will care, and they will have to read the technical manuals anyway. The style of the manual is informal mostly because I have never heard a good explaination for instructions that are written in a cold manner. We bought our computers to learn and have fun!.

HOW TO LOAD

TAPE....

Place the tape in your recorder, label side up, and insert your BASIC Cartridge. Make sure the tape is rewound, and reset the counter to zero. Push PLAY on the recorder, type RUN"C:", and press RETURN. If the program won't start to load, try positioning the tape forwards or backwards a little. The easiest way to find the beginning is to listen to the "noise" on the tape with a regular recorder. When you find the steady tone, you have the beginning of the program. We recommend you write down the number on your recorder's counter as each program starts, this will make it easier to find each part later on.

DISK....

To load and run the disk, first turn on your disk drive. When the busy light goes out, place the disk into the drive, with the BASIC Cartridge in place. The program will load each part and run by itself.

Any defective tapes or disks should be returned to:
Educational Software inc.
4565 Cherryvale
Soquel, CA 95073
(408) 476-4901

What is Page Flipping

The idea of flipping pages is somewhat unique in small home computers. You're lucky, the ATARI just happens to be able to do page flipping. The reason for this is that the ATARI allows both its DISPLAY LIST and DISPLAY DATA to be stored in any free area of memory. A Display List is the small set of instructions every computer needs to tell itself how to put the display data on the screen. This is especially important for us to learn since the ATARI computers offer so many types of graphics modes. This special capability means that you can store the information for numerous pages (TV screens) of graphics and/or text and then later, in your program, go to any of these pages instantly. This makes things look much more professional.

—BASICS—

First, we are going to cover some basics. Memory is stored in something called RAM, which stands for random access memory. This is because you can either read or write to any place within it. Usually discussion of memory is in terms of bytes. For example, your memory(RAM) may consist of 16k,24k,32k,40k, or 48k bytes. Remember that 1k is actually 1024 bytes. Confused? GOOD! Then you may appreciate the fact that to flip screens of data we are going to just move down in memory by increments of 256 bytes at a time. These increments are called "pages" (ain't life simple???). So 4 pages is 4*256 = 1024 bytes = 1k . The last sentence is all you have to remember. Four pages of memory =1k bytes of memory. Oh yes, I should tell you that "k" is the symbol for 1000.

The term flipping pages would now become confusing - we are NOT flipping 256 bytes at a time. For this reason we will refer to the process as flipping screens or displays,ie., the stuff you see on the TV screen at any one time.

There will be two methods shown in our examples. DON'T JUST VIEW THEM. COPY THEM TO ANOTHER DISK OR TAPE AND MODIFY THEM FOR YOUR OWN USE. WE WANT YOU TO COPY OUR SOFTWARE FOR USE IN YOUR OWN PROGRAMS. EACH EXAMPLE IS SET UP FOR GENERAL PURPOSES, SO JUST SUBSTITUTE YOUR OWN TEXT OR GRAPHICS. IF YOU HAVE 32K OR MORE OF MEMORY, CHANGE TO HIGH RES, GRAPHICS MODE 8 AND ADD TEXT AS WELL AS GRAPHICS. The method and examples all work, but could be much better....especially if written for more memory than the small amount this lesson runs on (don't brag about your 48k machine. My ATARI has 160k!)

— METHOD 1 —

Normally when you say Graphics 0 to 8 (11 for newer units) to the computer, it sets up both a Display List and a data area just below the top of your memory. Method one simply tells the computer after a first screen is drawn, "memory ain't where it used to be, but it's now lower, so set up a new (additional) Display List and data area lower in memory". By doing this as often as you require (and memory space allows), you allow a number of screens full of data to be seen by just redirecting the one location that points to the start of each Display List. EASY! Figure one will show you this graphically:

TOP OF MEMORY

SCREEN DATA 1 (261 to 7900 bytes)
DISPLAY LIST (20 to 202 bytes)
SCREEN DATA 2
DISPLAY LIST 2
SCREEN DATA 3
DISPLAY LIST 3
AS MANY MORE SCREENS AS YOU HAVE ROOM FOR.
YOUR BASIC PROGRAM and OTHER STUFF.

BOTTOM OF MEMORY

Fig. 1

EXAMPLE 1

Run Example 1. Notice that you can see it draw out some random lines, then they disappear and a new set of lines is drawn out. Then...these two screens seem to appear and disappear, first one then another.
Here's how it works:

```
10 GRAPHICS 6
12 GOSUB 4000
15 TRAP 20
20 COLOR 1:FOR I=1 TO 20:COLOR 2*RND(4):DRAWTO 140*RND(4),70
   *RND(9):NEXT I
25 ? "THIS IS FLIPPING BETWEEN TWO AREAS OF MEMORY.  PRESS
   RETURN TO CONTINUE."
30 A=PEEK(106)
40 DLISTL1=PEEK(560):DLISTH1=PEEK(561)
50 POKE 106,A-32
60 GRAPHICS 6
61 GOSUB 4000
70 DLISTL2=PEEK(560):DLISTH2=PEEK(561)
75 TRAP 80
80 COLOR 1:FOR I=1 TO 20:COLOR 2*RND(4):DRAWTO 140*RND(4),70
   *RND(9):NEXT I
85 ? "THIS IS FLIPPING BETWEEN TWO AREAS OF MEMORY.  PRESS
   RETURN TO CONTINUE."
86 POKE 764,255
90 POKE 561,DLISTH1:FOR W=1 TO 2:NEXT W:POKE 561,DLISTH2:IF
   PEEK(764)=12 THEN 110
100 GOTO 90
110 POKE 106,A:RUN "D:NEXT1"
4000 X=PEEK(16):IF X=128 THEN 4020
4010 POKE 16,X-128:POKE 53774,X-128
4020 RETURN
```

Lines 10 to 25 draw a screen just like normal using COLOR, PLOT, and PRINT commands (See your BASIC referance manual for these commands).

Line 30 stores the original top of memory (in number of 256 byte pages) read from location 106 using the PEEK command.

Line 40 stores the two numbers that locate the Display List. Remember that it takes two eight bit numbers to address all of the computers memory. The LOW part can hold from 0 to 255 and the HIGH part holds from 0 to 255 (times 256)...again see your manual or our MASTER MEMORY MAP. We will only use the part of the address in location 561. This is because we are moving memory down in whole page increments and 561 holds the number of whole pages. We left in the low part, in case you want to experiment with your own ideas.

— NOTE —

Often I leave out pieces of code that you may not see mentioned in the manual. You should assume that it either is something you will see the need for later, or something needed just to make all the examples run together smoothly.

Line 50 says "the top of memory is located down in RAM by 32 pages (8k bytes) from the previous value. This is more than is needed for this example. The extra amount will allow you room to add more screens later when you come back to this example to experiment with your own changes.... and you will come back, won't you?

Line 60 to 85 again draw some graphics just like normal. The computer looks at location 106 and sees the value we put there. It is thus "fooled" into placing the new Display List and data starting 32 pages lower than it otherwise would have. Actually,the amount of pages you go down should correspond to the amount of memory the graphics modes you are using requires. Look in the first page of the BASIC Manual's Graphics section to see how much you need. We used mode 6, so 2k or 8 pages would have been enough. This way, though, we left room for you to easily copy this program and substitute your own graphics!

Line 70 stores the location or address of this second Display List. You can do this before (like here, line 70) or after the graphics statements. The location will be used to flip the screens later.

Line 90 then does the flipping by just POKEing 561 with the address (high part only!) of the first Display List, then the second, then the first, etc. This says "use this Display List..no this one...no use the first..etc.".

ANOTHER NOTE...

You can see in the program code in Example 1 it is line 110 that POKEs 106 back to the original value. If you are going to go on and use your computer after confusing it like this, you had better tell it what the real top of memory is. If, on the other hand, you are going to turn it off after our program is finished, don't bother. Any time you power up or press RESET, the computer will store the correct value in 106.

Also, while running our program, don't press RESET or BREAK unless it crashes (it shouldn't, but...). If any example stops then press RESET and type RUN and press RETURN. Of course, you should press RESET and then experiment with each example after you have seen the entire program at least once. This is how you will learn!

EXAMPLE 2

This example is made out of Example 1 to show you how to expand the basic idea of changing screens. Here we set up FOUR Display Lists and print four simple messages on the screen. As you press a button the message appears instantly! If you think about it, you will realize that there is no diference in flipping between four one line messages on a page, or four completely filled pages of text. Each time you press a button, another area of memory is displayed on the screen. The one line messages were just to keep the program simple for you to study. Please add many more lines of text to this example and see what we mean. You just put more words between the quotes in lines 20, 70, 110, & 150.

Lines 10 to 40 write text and store the needed values for Display List one (DL1).

Lines 50 to 80 POKEs 106 down 8 pages (2k, more than enough for GR.0), writes more text, and stores these new values.

Lines 90 to 120 same thing another 8 pages down in memory, but different text.

Lines 130 to 160 same thing again for a fourth screen!

Lines 168 to 185 look for you to press keys 1 to 4 (or whatever!) so that lines 190 to 220 can tell the computer which of the four Display List you previously created to now use. The test for CH=12 is to see if you decided to go on to something else by pressing return. The POKE of 255 to 764 clears the location that holds the internal code for the last key pressed. The variable CH on line 170 will tell the system which key you just pressed (if any).

NOTE

You can put in your own text or have the computer read in the text from the keyboard or disk and place the first 960 bytes (the size of GR.0) in page one, the next 960 in page 2, etc. The way to read in data from keyboard or disk is explained in your reference manuals.

```
5 TRAP 10
10 GRAPHICS 0
20 POSITION 4,10:? "THIS IS PAGE 1.":POSITION 2,20:? "PRESS
   1,2,3 OR 4 FOR THAT PAGE."
25 ? "PRESS RETURN TO CONTINUE."
30 A=PEEK(106)
```

```
40 DLL1=PEEK(560):DLH1=PEEK(561)
45 REM ****************************
50 POKE 106,A-8
60 GRAPHICS 0
70 POSITION 10,10:? "THIS IS PAGE 2.":POSITION 2,20:? "PRESS
   1,2,3 OR 4 FOR THAT PAGE."
75 ? "PRESS RETURN TO CONTINUE."
80 DLL2=PEEK(560):DLH2=PEEK(561)
85 REM ****************************
90 POKE 106,A-16
100 GRAPHICS 0
110 POSITION 15,10:? "THIS IS PAGE 3.":POSITION 2,20:? "PRESS
    1,2,3 OR 4 FOR THAT PAGE."
115 ? "PRESS RETURN TO CONTINUE."
120 DLL3=PEEK(560):DLH3=PEEK(561)
125 REM ****************************
130 POKE 106,A-24
140 GRAPHICS 0
150 POSITION 20,10:? "THIS IS PAGE 4.":POSITION 2,20:? "PRESS
    1,2,3 OR 4 FOR THAT PAGE."
155 ? "PRESS RETURN TO CONTINUE."
160 DLL4=PEEK(560):DLH4=PEEK(561)
165 REM ****************************
168 POKE 764,255:GOSUB 4000
170 CH=PEEK(764)
180 IF CH=31 THEN 190
181 IF CH=30 THEN 200
182 IF CH=26 THEN 210
183 IF CH=24 THEN 220
184 IF CH=12 THEN 230
185 GOTO 170
190 POKE 560,DLL1:POKE 561,DLH1:GOTO 170
200 POKE 560,DLL2:POKE 561,DLH2:GOTO 170
210 POKE 560,DLL3:POKE 561,DLH3:GOTO 170
220 POKE 560,DLL4:POKE 561,DLH4:GOTO 170
230 POKE 106,A:RUN "D:NEXT2"
4000 X=PEEK(16):IF X=128 THEN 4020
4010 POKE 16,X-128:POKE 53774,X-128
4020 RETURN
```

EXAMPLE 3

Example 3 is the same as Example 2, but with graphics instead of text. Although it will seem obvious to some of you, all you need to do is substitute graphics type commands for the text commands of Example 2. We used four simple bar graphs, but you could draw very complicated pictures if you wanted. Remember that the material that you place on a screen doesn't effect the basic method we are using. Even if

you fill up the screen, the computer just keeps looking at the Display List to see where to get it's screen data. No more watching complicated pictures drawn out every time you need them. Now you can store them ahead of time in memory.

```
5 TRAP 10
10 GRAPHICS 5
15 COLOR 1:PLOT 5,5:DRAWTO 5,35:DRAWTO 75,35
20 COLOR 2:PLOT 10,34:DRAWTO 10,25:DRAWTO 5,25:POSITION 5,34
   :POKE 765,2:XIO 18,#6,0,0,"S:"
25 ? "PRESS 1,2,3, OR 4 FOR BAR GRAPHS OF":? "THE YEARS 1981,
   1982,1983 OR 1984.":? "PRESS RETURN TO GO ON."
30 A=PEEK(106)
40 DLL1=PEEK(560):DLH1=PEEK(561)
45 REM ******************************
50 POKE 106,A-8
60 GRAPHICS 5
65 COLOR 1:PLOT 5,5:DRAWTO 5,35:DRAWTO 75,35
70 COLOR 3:PLOT 15,34:DRAWTO 15,20:DRAWTO 10,20:POSITION 10,
   34:POKE 765,3:XIO 18,#6,0,0,"S:"
75 ? "PRESS 1,2,3, OR 4 FOR BAR GRAPHS OF":? "THE YEARS 1981,
   1982,1983 OR 1984.":? "PRESS RETURN TO GO ON."
80 DLL2=PEEK(560):DLH2=PEEK(561)
85 REM ******************************
90 POKE 106,A-16
100 GRAPHICS 5
105 COLOR 1:PLOT 5,5:DRAWTO 5,35:DRAWTO 75,35
110 COLOR 1:PLOT 20,34:DRAWTO 20,15:DRAWTO 15,15:POSITION 15,
    34:POKE 765,1:XIO 18,#6,0,0,"S:"
115 ? "PRESS 1,2,3, OR 4 FOR BAR GRAPHS OF":? "THE YEARS 1981,
    1982,1983 OR 1984.":? "PRESS RETURN TO GO ON."
120 DLL3=PEEK(560):DLH3=PEEK(561)
125 REM ******************************
130 POKE 106,A-24
140 GRAPHICS 5
145 COLOR 1:PLOT 5,5:DRAWTO 5,35:DRAWTO 75,35
150 COLOR 2:PLOT 25,34:DRAWTO 25,10:DRAWTO 20,10:POSITION 20,
    34:POKE 765,2:XIO 18,#6,0,0,"S:"
155 ? "PRESS 1,2,3, OR 4 FOR BAR GRAPHS OF":? "THE YEARS 1981,
    1982,1983 OR 1984.":? "PRESS RETURN TO GO ON."
160 DLL4=PEEK(560):DLH4=PEEK(561)
165 REM ******************************
168 POKE 764,255:GOSUB 4000
170 CH=PEEK(764)
180 IF CH=31 THEN 190
181 IF CH=30 THEN 200
182 IF CH=26 THEN 210
183 IF CH=24 THEN 220
184 IF CH=12 THEN 230
185 GOTO 170
```

```
190 POKE 560,DLL1:POKE 561,DLH1:GOTO 170
200 POKE 560,DLL2:POKE 561,DLH2:GOTO 170
210 POKE 560,DLL3:POKE 561,DLH3:GOTO 170
220 POKE 560,DLL4:POKE 561,DLH4:GOTO 170
230 POKE 106,A:RUN "D:NEXT3"
4000 X=PEEK(16):IF X=128 THEN 4020
4010 POKE 16,X-128:POKE 53774,X-128
4020 RETURN
```

EXAMPLE 4

This example draws a shape on each page. Then, by flipping screens you can animate the shape! Remember that any set of data can be used for the shape so why not copy this program to your disk/cassette and try your own shapes. Another idea would be to draw a business logo and move it across a chart of profits. Use your imagination, or just play if you like. The posibilities are endless!

```
3 GRAPHICS 0
5 TRAP 10
10 GRAPHICS 5
15 COLOR 1
20 READ X,Y:IF X=0 THEN 30
25 PLOT X,Y:GOTO 20
27 DATA 9,23,10,23,11,23,9,24,10,24,11,24,10,25,8,26,9,26,
   10,26,11,26,12,26,7,27,9,27,10,27,11,27,13,27
28 DATA 6,28,9,28,10,28,11,28,14,28,9,29,10,29,11,29,9,30,
   10,30,11,30,9,31,11,31,8,32,12,32,7,33,13,33,7,34,13,34
29 DATA 7,35,8,35,9,35,13,35,14,35,15,35,0,0,0
30 A=PEEK(106)
35 ? "     HUP!"
40 DLL1=PEEK(560):DLH1=PEEK(561)
45 REM ******************************
50 POKE 106,A-8
60 GRAPHICS 5
63 COLOR 1:RESTORE 77
65 ? "     TWO!"
70 READ X,Y:IF X=0 THEN 80
75 PLOT X,Y:GOTO 70
77 DATA 19,23,20,23,21,23,19,24,20,24,21,24,20,25,18,26,19,
   26,20,26,21,26,22,26,17,27,19,27,20,27,21,27,23,27
78 DATA 17,28,19,28,20,28,21,28,23,28,19,29,20,29,21,29,19,
   30,20,30,21,30,19,31,21,31,19,32,21,32,18,33,22,33
79 DATA 18,34,22,34,18,35,19,35,20,35,22,35,23,35,24,35,0,0
80 DLL2=PEEK(560):DLH2=PEEK(561)
85 REM ******************************
```

```
90 POKE 106,A-16
100 GRAPHICS 5
103 ? "     THREE!"
105 COLOR 1:RESTORE 117
110 READ X,Y:IF X=0 THEN 120
115 PLOT X+20,Y:GOTO 110
117 DATA 9,23,10,23,11,23,9,24,10,24,11,24,10,25,8,26,9,26,
    10,26,11,26,12,26,7,27,9,27,10,27,11,27,13,27
118 DATA 6,28,9,28,10,28,11,28,14,28,9,29,10,29,11,29,9,30,
    10,30,11,30,9,31,11,31,8,32,12,32,7,33,13,33,7,34,13,34
119 DATA 7,35,8,35,9,35,13,35,14,35,15,35,0,0,0
120 DLL3=PEEK(560):DLH3=PEEK(561)
125 REM ****************************
130 POKE 106,A-24
140 GRAPHICS 5
143 ? "     FOUR!"
145 COLOR 1:RESTORE 157
150 READ X,Y:IF X=0 THEN 160
155 PLOT X+20,Y:GOTO 150
157 DATA 19,23,20,23,21,23,19,24,20,24,21,24,20,25,18,26,19,
    26,20,26,21,26,22,26,17,27,19,27,20,27,21,27,23,27
158 DATA 17,28,19,28,20,28,21,28,23,28,19,29,20,29,21,29,19,
    30,20,30,21,30,19,31,21,31,19,32,21,32,18,33,22,33
159 DATA 18,34,22,34,18,35,19,35,20,35,22,35,23,35,24,35,0,0
160 DLL4=PEEK(560):DLH4=PEEK(561)
165 REM ****************************
168 POKE 764,255
170 CH=PEEK(764)
180 IF CH=31 THEN 190
181 IF CH=30 THEN 200
182 IF CH=26 THEN 210
183 IF CH=24 THEN 220
184 IF CH=12 THEN 230
185 GOTO 170
190 POKE 560,DLL1:POKE 561,DLH1:GOTO 170
200 POKE 560,DLL2:POKE 561,DLH2:GOTO 170
210 POKE 560,DLL3:POKE 561,DLH3:GOTO 170
220 POKE 560,DLL4:POKE 561,DLH4:GOTO 170
230 POKE 106,A:RUN "D:NEXT4"
```

EXAMPLE 5

Notice line 7 which stores the value held in memory location 559. Then line 13 POKEs 559 with a 0. Well, this neat trick turns off the screen so that the pictures are drawn without your seeing them. It also speeds up the computer by about 30% depending on graphics mode. Want to know why? Send us $40. and we..., oh well, I'll tell you. Location 559 controls the ANTIC Chip which puts the video on

the screen. Use the original value, stored in "NON" to turn it on. Use 0 to turn it off. Line 169 is where we turn the display back on.

THIS WORKS FOR ANY PROGRAM!

```
5 TRAP 10
7 NON=PEEK(559)
10 GRAPHICS 5
13 POKE 559,0
15 COLOR 1
20 READ X,Y:IF X=0 THEN 30
25 PLOT X,Y:GOTO 20
27 DATA 9,23,10,23,11,23,9,24,10,24,11,24,10,25,8,26,9,26,
   10,26,11,26,12,26,7,27,9,27,10,27,11,27,13,27
28 DATA 6,28,9,28,10,28,11,28,14,28,9,29,10,29,11,29,9,30,
   10,30,11,30,9,31,11,31,8,32,12,32,7,33,13,33,7,34,13,34
29 DATA 7,35,8,35,9,35,13,35,14,35,15,35,0,0,0
30 A=PEEK(106)
35 ? "     HUP!"
40 DLL1=PEEK(560):DLH1=PEEK(561)
45 REM ******************************
50 POKE 106,A-8
60 GRAPHICS 5
62 POKE 559,0
63 COLOR 1:RESTORE 77
65 ? "     TWO!"
70 READ X,Y:IF X=0 THEN 80
75 PLOT X,Y:GOTO 70
77 DATA 19,23,20,23,21,23,19,24,20,24,21,24,20,25,18,26,19,
   26,20,26,21,26,22,26,17,27,19,27,20,27,21,27,23,27
78 DATA 17,28,19,28,20,28,21,28,23,28,19,29,20,29,21,29,19,
   30,20,30,21,30,19,31,21,31,19,32,21,32,18,33,22,33
79 DATA 18,34,22,34,18,35,19,35,20,35,22,35,23,35,24,35,0,0
80 DLL2=PEEK(560):DLH2=PEEK(561)
85 REM ******************************
90 POKE 106,A-16
100 GRAPHICS 5
102 POKE 559,0
103 ? "     THREE!"
105 COLOR 1:RESTORE 117
110 READ X,Y:IF X=0 THEN 120
115 PLOT X+20,Y:GOTO 110
117 DATA 9,23,10,23,11,23,9,24,10,24,11,24,10,25,8,26,9,26,
    10,26,11,26,12,26,7,27,9,27,10,27,11,27,13,27
118 DATA 6,28,9,28,10,28,11,28,14,28,9,29,10,29,11,29,9,30,
    10,30,11,30,9,31,11,31,8,32,12,32,7,33,13,33,7,34,13,34
119 DATA 7,35,8,35,9,35,13,35,14,35,15,35,0,0,0
120 DLL3=PEEK(560):DLH3=PEEK(561)
125 REM ******************************
```

```
130 POKE 106,A-24
140 GRAPHICS 5
142 POKE 559,0
143 ? "     FOUR!"
145 COLOR 1:RESTORE 157
150 READ X,Y:IF X=0 THEN 160
155 PLOT X+20,Y:GOTO 150
157 DATA 19,23,20,23,21,23,19,24,20,24,21,24,20,25,18,26,19,
    26,20,26,21,26,22,26,17,27,19,27,20,27,21,27,23,27
158 DATA 17,28,19,28,20,28,21,28,23,28,19,29,20,29,21,29,19,
    30,20,30,21,30,19,31,21,31,19,32,21,32,18,33,22,33
159 DATA 18,34,22,34,18,35,19,35,20,35,22,35,23,35,24,35,0,0
160 DLL4=PEEK(560):DLH4=PEEK(561)
165 REM ****************************
168 POKE 764,255
169 POKE 559,NON
170 CH=PEEK(764)
180 IF CH=31 THEN 190
181 IF CH=30 THEN 200
182 IF CH=26 THEN 210
183 IF CH=24 THEN 220
184 IF CH=12 THEN 230
185 GOTO 170
190 POKE 560,DLL1:POKE 561,DLH1:GOTO 170
200 POKE 560,DLL2:POKE 561,DLH2:GOTO 170
210 POKE 560,DLL3:POKE 561,DLH3:GOTO 170
220 POKE 560,DLL4:POKE 561,DLH4:GOTO 170
230 POKE 106,A:RUN "D:NEXT5"
```

METHOD 2

This method differs only slightly from the first, but allows you more control of what goes on. The examples will explain the differences.

EXAMPLE 6

Instead of using a variable called "A", we use "P106" to store the original # of pages in your memory. This will be more meaningful to us. P106 stands for the value to POKE into location 106.

```
4 P106=PEEK(106)
5 ? "†":? "AT PAGE ONE!":? "PRESS 1 OR 2 FOR THAT PAGE."
7 ? "PRESS RETURN TO GO ON."
```

```
10 DP=PEEK(560)+PEEK(561)*256
12 POKE 16,64
15 SAV=PEEK(DP+5)
16 POKE 106,P106-4:POKE 89,SAV-4
17 ? "AT PAGE TWO!":? "PRESS 1 OR 2 FOR THAT PAGE.":? "PRESS
   RETURN TO GO ON."
30 POKE 764,255:TRAP 80
33 CH=PEEK(764)
35 IF CH=31 THEN 45
36 IF CH=12 THEN 60
37 IF CH=30 THEN 40
38 GOTO 33
40 POKE DP+5,SAV-4
43 GOTO 33
45 POKE DP+5,SAV
55 GOTO 33
60 POKE 106,P106:RUN "D:NEXT6"
```

Line 10 stores the location of the start of the DL as one decimal number. Line 15 store the number we are after. It comes 5 bytes after the start of the DL, so we PEEK at DL+5, ie. it is the sixth number in the DL.

Line 16 POKEs memory location 106 down by a number of pages (4 in this case). This line also stores a new value we need: location 89. This one is a copy of the value in DP+5 which tells the system where the start of the display data is. The computer looks at the DL whenever a GRAPHICS command is used, and stores that value here so that it will know where the start of your data is. After a Graphics call, we are free to change this number (in 89) to "fool" the computer into doing what we want. After POKEing both of these locations down far enough, we now write some text to the new area of memory on line 17. This could be done many times if you have enough RAM.

Now when lines 30 to 38 choose which screen you want, lines 40 or 45 just change the value in the first Display List that controls where the first DL gets it's data from. We don't care about a second (or 3rd or 4th...) Display List as in Example 1.

SO WHAT?

Well, now by just changing one value at the start of the first Display List+5 (DP+5), you can tell the system to go display different data from all over memory. This change could be easily controlled with a joystick as we do in our Scrolling Program, Tricky Tutorial #2.

Also, you might want to look at one screen while you are drawing several others; for example while a decision was being made about options on the first.

EXAMPLE 7

This does what we just suggested. You can not only look at two screens (press 0 for screen 1 and 4 for screen two), but by inputing other positive numbers you can look down in memory. By inputing negative numbers, you look up in memory until you reach the top. The stuff you see on the screen will be the alpha-numeric equivalent of the BASIC program, your screen data, the Operating System or whatever you are looking at. The only changes to this program are:

Lines 20 & 33 input a number.

Line 35 tests that number to see if it is too big.

Line 40 redirects the DL as before, but with your value instead of the previous fixed value of 4.

```
4 P106=PEEK(106)
5 ? "†":? "AT PAGE ONE!":? "PRESS START AND RETURN AT THE
  SAME    TIME TO GO ON.":? "PRESS BETWEEN 0 AND ";P106-5;
7 ? " TO LOOK AT    MEMORY IN 1/4 PAGE SCREEN INCREMENTS.":?
  "THEN PRESS RETURN. REPEAT AS DESIRED."
10 DP=PEEK(560)+PEEK(561)*256
12 POKE 16,64
15 SAV=PEEK(DP+5)
16 POKE 106,P106-4:POKE 89,SAV-4
17 ? "AT PAGE TWO!":? "   PRESS START AND RETURN AT THE SAME
   TIME TO GO ON"
20 DIM A(2)
25 POKE 53279,8
30 TRAP 30:Z=PEEK(53279):IF Z=6 THEN 80
33 INPUT A
35 IF A>(P106) THEN 60
40 POKE DP+5,SAV-A
55 GOTO 30
60 ? "NUMBER TOO LARGE, MUST BE LESS THAN";P106-5:? "WE ARE
   NOW AT 4 PAGES DOWN IN MEMORY"
65 POKE DP+5,SAV-4:POKE 89,SAV-4
70 GOTO 30
80 TRAP 40000:POKE 106,P106:POKE 89,SAV:POKE DP+5,SAV:RUN
   "D:NEXT7"
```

EXAMPLE 8

The last example is exactly the same as Example 7, except it looks at memory using a colorful graphics viewpoint of the data there. Be sure to try negitive numbers on the last two examples also. Since the program is designed to look up or down in memory, the negitive numbers look down while the positives look higher in RAM. You can look most anywhere from within the BASIC Cartridge to the data flowing in and out of the Operating System (try large negative numbers for this)...See if you can find an area that is changing for real special effects.

```
4 P106=PEEK(106)
5 GRAPHICS 5:COLOR 1:PLOT 10,10:DRAWTO 10,20:DRAWTO 40,20:
  DRAWTO 40,10:DRAWTO 10,10
10 DP1=PEEK(560)+PEEK(561)*256
12 POKE 16,64
15 SAV1=PEEK(DP1+5)
16 POKE 106,P106-8
17 GRAPHICS 5:COLOR 2:PLOT 20,20:DRAWTO 20,30:DRAWTO 30,30:
   DRAWTO 30,20:DRAWTO 20,20
20 DIM A(2)
21 DP2=PEEK(560)+PEEK(561)*256
22 SAV2=PEEK(DP2+5)
25 POKE 53279,8
27 ? "PRESS START AND RETURN TOGETHER TO GO ON."
30 TRAP 30:Z=PEEK(53279):IF Z=6 THEN 80
33 INPUT A
35 IF A>(P106) THEN 60
40 REM POKE DP1+5,SAV-A
41 POKE DP2+5,SAV1-A
55 GOTO 30
60 ? "NUMBER TOO LARGE, MUST BE LESS THAN";P106-5:? "WE ARE
   NOW AT 4 PAGES DOWN IN MEMORY."
65 POKE DP+5,SAV-4:POKE 89,SAV-4
70 GOTO 30
80 TRAP 40000:POKE 106,P106:POKE 89,SAV:POKE DP+5,SAV:RUN
   "D:NEXT8"
```

THAT'S IT

Just take any of our examples and try to modify them so that you can both understand the methods and make your programs look and run much cleaner. I hope you find many new uses for PAGE FLIPPING. Please write and tell me about your accomplishments using techniques in TRICKY TUTORIALS. BYE!!